LABORATORIO 6 – Básico

Programas Ada con Vectores, Matrices y Registros

1. Implementa la función **esta_en_vector** que, dado un valor entero y un vector de N enteros (completo, o sea, con N enteros) cuyos valores pueden no estar ordenados, diga si el entero está o no en el vector.
2. Implementa la función **posición** que, dado un valor entero X y un vector de N enteros (completo, o sea, con N enteros) cuyos valores pueden no estar ordenados, diga en qué posición se encuentra X en el vector. Si dicho entero estuviera repetido, bastará con devolver la posición de una cualquiera de sus apariciones; y en caso de no encontrarse en el vector, se devolverá el valor Integer'Last.
3. Implementa la función **son_iguales** que, dadas dos fechas, nos indica si se trata de la misma fecha o no.
4. Implementa la función **centesimales** que, dado un valor de grados sexagesimales (grados, minutos y segundos), los devuelva los grados centesimales equivalentes.
5. Implementa la función **son_iguales** que, dadas dos listas incluidas en un vector (terminado en un centinela), devuelva true si el contenido de las dos listas es el mismo (mismos elementos en el mismo orden) y false si no.
6. Crear el procedimiento **eliminar_elemento** que, dada una posición pos y un vector V de N enteros (no necesariamente completo, o sea, con, como mucho, N enteros terminado con el valor Integer'First) cuyos valores pueden no estar ordenados, modifique V eliminando el elemento en la posición pos (si existe). Si pos no es un valor donde hay un elemento, entonces no se hará nada. Los elementos de la lista no tienen por qué estar ordenados (ni antes ni después del borrado), lo que se puede aprovechar para ganar en eficiencia al eliminar el elemento. De hecho, es imprescindible que la solución sea eficientey, en concreto, se aprovechará que el orden de los elementos de la lista es indiferente (tanto antes como después de la eliminación del elemento) para borrar el tercer elemento sin utilizar bucles.
7. Implementa el procedimiento **insertar_elemento_en_pos** que, dados dos enteros *num* y *pos,* y un vector de, como máximo N-1 enteros (lo que significa que aún hay hueco para uno más) y terminado con el valor Integer'First, inserte en la lista el número *num* en la posición *pos*, desplazando los elementos que corresponda una posición a la derecha. Para resolver este problema conviene reutilizar el programa *rotar_derecha.* Aunque se puede usar sin modificar, quizás la manera más intuitiva sea haciéndole una pequeña modificación y añadiéndole como parámetros de entrada la posición *Pos* y el número a insertar *Num*.
8. Implementa la función **pos_maximo** que, dado un vector de enteros, devuelva la posición en la que se encuentra el máximo de ese vector.
9. Implementa el procedimiento **ordenar_seleccion** que, dado un vector de enteros, ordene sus elementos de mayor a menor. El programa debe usar la función maximo del punto anterior.

LABORATORIO 6 – Avanzado

Programas Ada con Vectores y Registros

1. Implementa el procedimiento **cent_a_sex** que, dado un valor de grados centesimales, lo convierta a grados sexagesimales (grados, minutos y segundos).
2. Crear el procedimiento **rotar_derecha** que, dado un vector de N enteros (completo, o sea, con N enteros) cuyos valores pueden no estar ordenados, lo modifique rotando sus elementos una posición a la derecha, de modo que el último elemento pase a ser el primero, tal y como se muestra en el siguiente caso de prueba:

	1	2	3	4	5	6	7	8	9	10
Vector de entrada	67	45	23	78	12	40	55	24	89	34
Vector de salida	34	67	45	23	78	12	40	55	24	89

3. Implementa la función **esta_en_vector_ordenado** que, dado un valor entero y un vector de N enteros (completo, o sea, con N enteros) cuyos valores están ordenados ascendentemente, diga si el entero está o no en el vector. Es imprescindible que la solución sea eficiente. En concreto, no se deberá seguir buscando el número cuando se sepa objetivamente que ya no puede estar en el vector (porque éste está ordenado).
4. Crear el procedimiento **eliminar_elemento_ordenado** que, dada una posición pos y un vector V de N enteros cuyos valores están ordenados ascendentemente y terminado con el valor Integer'First, modifique V eliminando el elemento en la posición pos (si existe). Si pos no es un valor donde hay un elemento, entonces no se hará nada. Este ejercicio es similar al anterior, con la diferencia de que en este caso los elementos de la lista están ordenados antes y deben seguir ordenados después de realizar el borrado del elemento.
5. Implementa el procedimiento **insertar_elemento_ordenado** que, dado un entero *num,* y un vector de, como máximo N-1 enteros (lo que significa que aún hay hueco para uno más) ordenados ascendentemente y terminado con el valor Integer'First, inserte en el vector el número *num* en la posición que le corresponda, desplazando los elementos mayores que él una posición a la derecha.
6. Implementa el procedimiento **eliminar_repetidos** que, dado un vector de enteros, lo modifique eliminando los enteros repetidos (borrando el más lejano al principio del vector) y haciendo que el vector termine en Integer'First.
7. Implementa la función **letra_mas_repetida** que, dado un vector de caracteres, cuente la frecuencia de las letras (sin distinguir entre mayúsculas y minúsculas) en ese vector y devuelva la más frecuente de todas. Si hubiera varias más frecuentes, la primera por orden alfabético.
8. Implementa el procedimiento **ordenar_insercion** que, dado un vector de enteros, ordene sus elementos de menor a mayor. Para hacer este procedimiento debes usar el procedimiento insertar_elemento_ordenado.
9. Implementa la función **es_Toeplitz** que, dada una matriz devuelva true si la matriz es una matriz de Töplitz. Se trata de una matriz cuadrada en la que los elementos de sus diagonales (de izquierda a derecha) son constantes. Una matriz de Toeplitz presenta la siguiente estructura:

$$\begin{pmatrix} a & b & c & d & e \\ f & a & b & c & d \\ g & f & a & b & c \\ h & g & f & a & b \\ i & h & g & f & a \end{pmatrix}$$